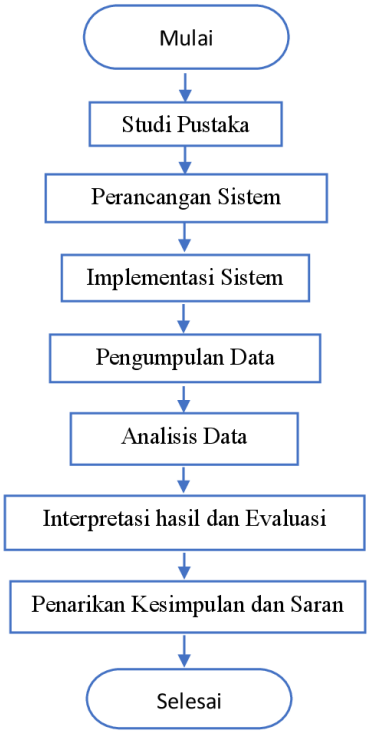
BAB III
METODE PENELITIAN

3.1 Desain Penelitian

Penelitian menggunakan pendekatan deskriptif kuantitatif, yaitu penelitian menganalisis data berupa angka pada *latency* dan biaya layanan *cloud*. Tujuannya untuk menggambarkan karakteristik dan performa layanan *serverless* AWS dan GCP berdasarkan data pengukuran yang nyata. Fokus pendekatan terdapat pada pengumpulan dan analisis data numerik untuk menguji serta menemukan pola hubungan antar variabel (Alford & Teater, 2025). Metode ini memungkinkan penerapan analisis statistik sehingga hasil penelitian dapat digeneralisasi ke populasi yang lebih luas (J. Branaghan dkk, 2021). Penelitian ini bersifat komparatif, karena tujuan utama penelitian adalah membandingkan dua objek layanan dalam aspek *latency* dan biaya dalam pemrosesan data *real-time* dari perangkat IoT.

3.2 Alur Penelitian

Alur penelitian yang mencakup serangkaian tahapan tahapan dari penelitian yang akan dilakukan. Tahapan dimulai dari studi pustaka untuk merumuskan masalah dan landasan teori, lalu dilanjutkan perancangan sistem sebagai acuan sebelum melakukan implementasi. Setelah sistem diimplementasikan, dilakukan pengumpulan data selama periode pengujian yang selanjutnya dianalisis untuk memperoleh data *latency* dan biaya layanan. Hasil analisis selanjutnya diinterpretasikan dan dievaluasi untuk menjawab tujuan dari penelitian yang dilakukan, terakhir penelitian ditutup dengan penarikan kesimpulan dan saran pengembangan. Secara visual, alur penelitian dapat dilihat pada gambar 3.1 berikut.



Gambar 3.1 Alur Penelitian

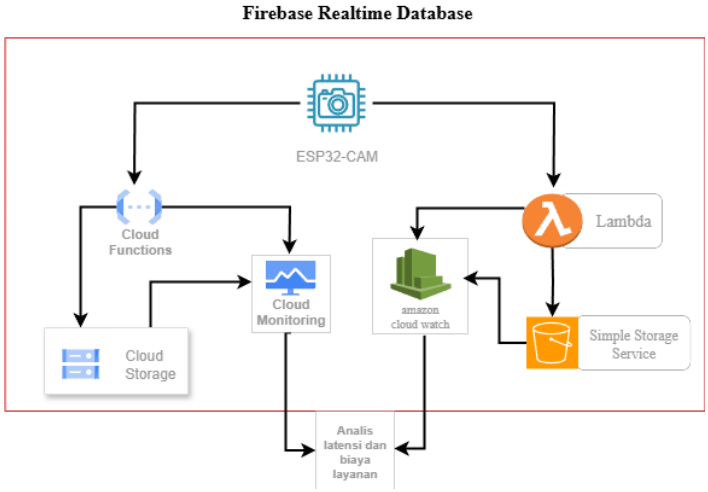
3.2.1 Studi Pustaka

Tahapan pertama dalam penelitian adalah melakukan studi pustaka yaitu mengumpulkan dan mempelajari berbagai literatur terkait perangkat IoT, *serverless computing*, serta arsitektur sistem *cloud* serta metode evaluasi performa layanan *cloud*. Literatur yang digunakan bersumber dari buku, jurnal ilmiah, prosiding konferensi, dan dokumentasi resmi layanan *cloud*. Studi pustaka ini bertujuan untuk membangun dasar teori yang kuat dan memahami konsep teknis yang relevan dengan sistem yang akan dikembangkan dalam penelitian.

3.2.2 Perancangan Sistem

Tahap kedua adalah perancangan sistem. Perancangan dilakukan dengan menyusun arsitektur sistem IoT berbasis perangkat ESP32-CAM sebagai pengirim data gambar. Sistem dirancang dengan dua skenario layanan *serverless*, yaitu AWS Lambda dan GCF sebagai penerima dan pemroses data dari perangkat IoT. Desain sistem juga mencakup konfigurasi *endpoint* API sebagai jalur pengiriman

data, integrasi dengan layanan penyimpanan *cloud* serta serta pencatatan data pada Firebase Realtime Database untuk mendukung pengumpulan data. Selengkapnya sistem dalam penelitian dapat dilihat pada gambar 3.2.



Gambar 3. 2 Rancangan Sistem

Pada sistem yang dikembangkan dalam penelitian ini, perangkat ESP32-CAM berfungsi sebagai pengirim data gambar secara periodik setiap 5 detik. Gambar yang diambil dikirimkan otomatis melalui HTTP *POST request* menuju *endpoint* API dari masing-masing layanan *cloud.* Terdapat dua jalur *endpoint* dalam sistem ini, yaitu, AWS API Gateway untuk jalur pemrosesan menggunakan AWS Lambda dan HTTP *Trigger* untuk jalur pemrosesan menggunakan GCF.

Setelah data gambar dikirim oleh perangkat, *endpoint* akan meneruskannya kepada fungsi *serverless* yang berjalan di masing-masing *platform*. Fungsi *serverless* pada AWS Lambda maupun GCF akan memproses data sesuai dengan kode yang diterapkan, kemudian mencatat hasil proses tersebut. Dalam penelitian ini, hasil pemrosesan dicatat dalam sistem log dan *monitoring* yang tersedia di masing-masing *platform cloud,* yaitu AWS CloudWatch Logs pada AWS dan Google Cloud Logging pada GCP, serta data pemrosesan juga dicatat ke Firebase Realtime Database.

Data yang tersimpan dalam log bawaan layanan *cloud* dan Realtime Database digunakan sebagai sumber data utama untuk analisis waktu *latency* dan

pencatatan aktivitas pemrosesan. Selain itu, data biaya penggunaan layanan *cloud* juga diambil dari *dashboard billing* masing-masing *platform* untuk dianalisis sebagai bagian dari pengukuran efisiensi layanan.

3.2.3 Implementasi Sistem

Pada tahap ketiga dilakukan implementasi teknis dari rancangan sistem yang telah dibuat. Implementasi mencakup pemrograman perangkat ESP32-CAM agar dapat menangkap gambar secara periodik dan mengirimkan data ke *endpoint* API layanan *cloud*. Selain itu, fungsi *serverless* diimplementasikan menggunakan bahasa pemrograman Python, lalu di*deploy* pada layanan *serverless*. Fungsi *serverless* diintegrasikan dengan Firebase Realtime Database untuk mencatat log pemrosesan dan data hasil pengujian secara otomatis. Proses konfigurasi integrasi dengan penyimpanan *cloud* serta pencatatan log juga dilakukan agar data yang dikirim dapat diproses dan dicatat secara otomatis. Tahapan awal dalam penelitian ini adalah melakukan persiapan lingkungan pengujian, yaitu sebagai berikut.

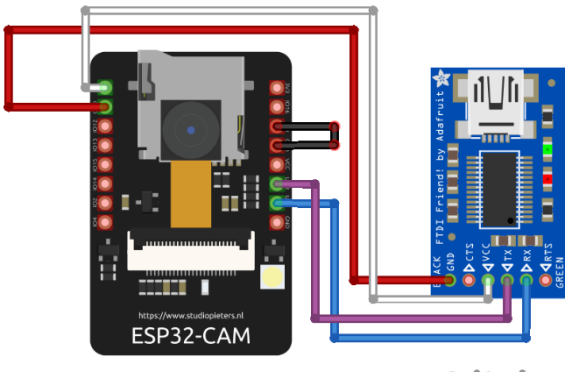
1. Mempersiapkan alat dan bahan yang digunakan dalam penelitian. Adapun alat dan bahan yang digunakan dapat dilihat pada tabel 3.1.

Tabel 3.1 Alat dan bahan penelitian

Perangkat Keras	Perangkat Lunak
1. ESP32-CAM 2. kamera OV2640 3. Modul FTDI 4. Laptop 5. Router untuk koneksi internet 6. Kabel USB 7. Kabel *Jumper female to female*	1. AWS Lambda 2. AWS CloudWatch 3. AWS Simple Storage Services 4. Google Cloud Functions 5. Google Cloud Logging 6. Firebase Realtime Database 7. *Dashboard Billing* AWS dan GCP 8. Arduino IDE 9. Spreadsheet/Google Sheets

2. Mengkonfigurasi ESP32-CAM agar dapat mengambil gambar secara otomatis setiap 5 detik dan mengirimkan gambar tersebut melalui HTTP

POST *request* ke masing-masing *endpoint* layanan *cloud*. Rangkaian dapat dilihat pada gambar 3.3 berikut.



Gambar 3.3 *Wiring* IoT

Wiring untuk menghubungkan modul ESP32-CAM ke laptop dapat dilakukan melalui FTDI programmer. Berikut koneksi pin yang dihubungkan dengan *jumper female to female*, dapat dilihat pada tabel.

Tabel 3.2 Koneksi pin ESP32-CAM x FTDI

ESP32-CAM	FTDI	Fungsi
U0T (TX)	RX	mengirim data dari ESP32-CAM ke FTDI
U0R (RX)	TX	mengirim data dari FTDI ke ESP32-CAM
GND	GND	*ground*
5V	VCC (5V)	sumber daya ke ESP32-CAM
IO0- GND		mode *flash programming* saat dihubungkan

Selanjutnya hubungkan FTDI ke laptop menggunakan kabel USB, lakukan unggah *firmware* di Arduino IDE pada *sketch* yang telah dibuat untuk program ESP32-CAM module AI Thinker ESP32-CAM. Setelah proses unggah selesai dan sudah terhubung, maka pin IO0-GND dapat di lepaskan, lalu klik tombol *reset* pada ESP32-CAM, maka *firmware* akan berjalan dan kamera mulai melakukan *snapshoot.*

3. Membuat akun layanan *cloud* pada AWS dan GCP
4. Membangun dan menerapkan fungsi *serverless* identik pada kedua *platform*:

a. *Deploy* fungsi di AWS Lambda menggunakan *API Gateway* sebagai *endpoint*.
b. *Deploy* fungsi di GCF menggunakan HTTP *Trigger* sebagai *endpoint.*

5. Memastikan seluruh sistem pencatatan data aktif dan terhubung:
   a. AWS CloudWatch Logs untuk pencatatan proses pada AWS Lambda.
   b. Google Cloud Logging untuk pencatatan proses pada GCF.
   c. Firebase Realtime Database sebagai media pencatatan terpusat data eksperimen dari ESP32-CAM ke layanan *cloud*.

Setelah seluruh komponen sistem siap dan dapat berkomunikasi dengan baik, dilakukan pengujian inti berupa pengiriman data gambar secara periodik dari ESP32-CAM ke AWS Lambda dan GCF selama periode pengujian. Data hasil pemrosesan dari kedua *platform* dicatat di sistem log dan *database* untuk dianalisis pada tahap berikutnya.

3.2.4 Pengumpulan Data

Tahap keempat dilakukan pengumpulan data setelah pengujian. Perangkat ESP32-CAM dikonfigurasi untuk mengirimkan data gambar secara periodik setiap 5000 milidetik (ms) selama 7 hari penuh. Data *latency* diperoleh dari log ESP32, AWS Lambda, GCF, Firebase Realtime Database, serta waktu penerimaan pada S3 dan GCS. Selain itu data dapat diperoleh melalui *dashboard billing* masing masing layanan. Semua data hasil eksperimen disimpan untuk dianalisis pada tahap berikutnya.

Dengan interval pengujian yang ditentukan, maka jumlah *request* teoritis adalah:

$$\mathrm{N} = \frac{\text{Durasi (detik)}}{\text{Interval (detik)}} \tag{3.1}$$

Karena durasi 1 hari maka $= \frac{24 \text{ jam} \times 60 \text{ menit} \times 60 \text{ detik}}{5} = 17.280$ *request*

Sehingga berdasarkan persamaan (3.1) total *request* teoritis selama 7 hari adalah:

$$17.280 \text{ x } 7 = 120.960 \textit{ request}$$

Untuk mengukur *latency*, dicatat *timestamp* pada 3 titik proses utama:

a. T1: *timestamp* saat ESP32-CAM mulai mengambil gambar dan mengirim data ke *endpoint* (AWS API Gateway atau HTTP *Trigger* GCF).
b. T2: *timestamp* saat data diterima dan mulai diproses oleh fungsi *serverless* (AWS Lambda atau GCF).
c. T3: Saat data tersimpan di AWS S3 atau GCS

Selain data *latency*, pengumpulan data biaya dilakukan dengan mengakses *dashboard billing* masing-masing *platform.* Berdasarkan hasil pengujian, hanya beberapa komponen layanan yang tercatat menghasilkan biaya, yaitu layanan penyimpanan objek melalui AWS S3 dan GCS, serta pencatatan log melalui AWS CloudWatch. Sementara itu, komponen lainnya seperti eksekusi fungsi *serverless* AWS Lambda, penggunaan API *Gateway*, serta HTTP *Trigger* tidak menghasilkan biaya selama masa pengujian karena penggunaan masih berada dalam batas layanan gratis. Oleh karena itu, analisis biaya difokuskan pada komponen-komponen yang benar-benar mencatat pengeluaran selama proses pengujian berlangsung. Selanjutnya biaya diakumulasikan per 1000 pemrosesan gambar.

3.2.5 Analisis Data

Tahap kelima adalah analisis data. Data yang telah terkumpul dari hasil pengujian diolah pada tahap analisis data. Hasil analisis disajikan dalam bentuk tabel dan grafik agar lebih mudah diinterpretasikan.

Latency end-to-end merupakan data yang dikirimkan oleh perangkat hingga tersimpan di *cloud storage*. Dapat didefinisikan sebagai selisih waktu dari proses pengambilan gambar hingga data berhasil tersimpan di *cloud storage*. *Latency* dihitung menggunakan rumus:

$$\textit{Latency}\ \text{End-to-End} = T3 - T1 \tag{3.2}$$

Perhitungan biaya dilakukan dari akumulasi pengeluaran aktual yang tercatat selama pengujian. Komponen biaya yang dianalisis adalah penyimpanan objek di AWS S3 dan GCS serta pencatatan log melalui AWS CloudWatch. Perhitungan biaya rata-rata per *request* dengan membagi total biaya terhadap jumlah *request* yang berhasil dikirim dan tercatat. Pendekatan ini digunakan untuk mendapatkan gambaran efisiensi biaya dari masing-masing *platform* dalam menangani permintaan dari perangkat IoT. Rumus biaya rata-rata per *request* dihitung dengan persamaan (3.3).

$$\text{Biaya Per } Request = \frac{Total\ Biaya}{Jumlah\ \text{Request}\ tercatat} \tag{3.3}$$

Setelah memperoleh biaya per *request,* selanjutnya diakumulasikan kedalam per 1000 data gambar, sehingga biaya layanan dari AWS dan biaya layanan dari GCP dapat dilakukan komparasi.

3.2.6 Interpretasi Hasil dan Evaluasi

Tahap berikutnya melakukan interpretasi hasil dan evaluasi, bertujuan untuk memahami dari perbedaan performa kedua layanan *cloud*. Pembahasan mencakup analisis keunggulan dan kelemahan masing-masing layanan pada aspek *latency* dan biaya pemrosesan, serta membandingkan temuan penelitian dengan literatur terkait. Evaluasi juga dilakukan untuk mengidentifikasi potensi perbaikan dari metode pengujian yang telah diterapkan.

3.2.7 Penarikan Kesimpulan dan Saran

Tahapan terakhir dalam penelitian ini adalah penarikan kesimpulan berdasarkan hasil interpretasi dan evaluasi, memuat pernyataan akhir terkait layanan *serverless* yang lebih optimal berdasarkan hasil pengujian. Selain itu, disusun saran yang dapat menjadi pertimbangan dalam pengembangan penelitian selanjutnya, baik terkait metode pengujian maupun eksplorasi layanan *cloud* lainnya di masa mendatang.